app.R

Yared Shumete

2022-04-04

```
#### Load packages ----
library(shiny)
```

```
## Warning: package 'shiny' was built under R version 4.1.3
```

```
library(shinythemes)
```

```
## Warning: package 'shinythemes' was built under R version 4.1.3
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(bslib)
```

```
## Warning: package 'bslib' was built under R version 4.1.3

##
## Attaching package: 'bslib'

## The following object is masked from 'package:utils':
##
##     page
```

```
#### Load data ----
# Read in PeterPaul processed dataset for nutrients.
# Specify the date column as a date
# Remove negative values for depth_id
```

```
# Include only lakename and sampledate through po4 columns

nutrient_data <- read.csv("Data/NTL-LTER_Lake_Nutrients_PeterPaul_Processed.csv")
nutrient_data$sampledate <- as.Date(nutrient_data$sampledate, format="%Y-%m-%d")
nutrient_data <-  nutrient_data %>%
  filter(depth_id >=0) %>%
  select(lakename, sampledate:po4)

#### Define UI ----

ui <- fluidPage(
    # Choose a title
  titlePanel("Lakes' Nutrient Concentration"),
    sidebarLayout(
     sidebarPanel(

      # Select nutrient to plot
      selectInput(inputId = "y",
                  label = "Nutrient",
                  choices = c("tn_ug", "tp_ug", "nh34", "no23", "po4"),
                  selected = "tn_ug"),

    # Select depth
    checkboxGroupInput(inputId = "x",
                       label = "Depth ID",
                       choices = unique(nutrient_data$depth_id),
                       selected = c(0, 7)),

    # Select lake
    checkboxGroupInput(inputId = "lakename",
                       label = "Lake",
                       choices = c("Paul Lake", "Peter Lake") ,
                       selected = "Paul Lake"),

    # Select date range to be plotted
    sliderInput(inputId = "date",
                label = "Year",
                min = as.Date("1991-05-20"),
                max = as.Date("2016-08-16"),
                value = c(as.Date("1991-05-20"), as.Date("2016-08-16")))),

  # Output: Description, lineplot, and reference
      mainPanel(
        plotOutput("scatterplot", brush = brushOpts(id = "scatterplot_brush")),
        tableOutput("mytable")
    )))

# theme = shinytheme("yeti"), it is not begin accepted
#### Define server  ----
server <- function(input, output) {

    # Define reactive formatting for filtering within columns
     filtered_nutrient_data <- reactive({
```

```
        nutrient_data %>%
          # Filter for dates in slider range
          filter(sampledate >= input$date[1] & sampledate <= input$date[2]) %>%
          # Filter for depth_id selected by user
          filter(depth_id %in% input$x) %>%
          # Filter for lakename selected by user
          filter(lakename %in% input$lakename)
      })

      # Create a ggplot object for the type of plot you have defined in the UI
        output$scatterplot <- renderPlot({
         ggplot(filtered_nutrient_data(),
                aes_string(x = "sampledate", y = input$y,
                           fill = "depth_id" , shape = "lakename")) +
           geom_point(alpha=0.8, size=2) +
           theme_classic(base_size = 14) +
           scale_shape_manual(values=c(21,24)) +
           labs(x = "Year" , y = expression(Concentration ~ (mu*g / L)),
                shape = "lakename", fill = "depth_id" ) +
           scale_fill_distiller(palette = "YlOrBr", guide = "colorbar", direction = 1)
           #scale_fill_viridis_c(option = "viridis", begin = 0, end = 0.8, direction = -1)
       })

     # Create a table that generates data for each point selected on the graph
        output$mytable <- renderTable({
          brush_out <- brushedPoints(filtered_nutrient_data(), input$scatterplot_brush)

        })

  }


#### Create the Shiny app object ----
shinyApp(ui = ui, server = server)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

```
#### Questions for coding challenge ----
#1. Play with changing the options on the sidebar.
    # Choose a shinytheme that you like. The default here is "yeti"
    # How do you change the default settings?
    # How does each type of widget differ in its code and how it references the dataframe?
#2. How is the mainPanel component of the UI structured?
    # How does the output appear based on this code?
#3. Explore the reactive formatting within the server.
    # Which variables need to have reactive formatting?
    # How does this relate to selecting rows vs. columns from the original data frame?
#4. Analyze the similarities and differences between ggplot code for a rendered vs. static plot.
    # Why are the aesthetics for x, y, fill, and shape formatted the way they are?
    # Note: the data frame has a "()" after it. This is necessary for reactive formatting.
    # Adjust the aesthetics, playing with different shapes, colors, fills, sizes, transparencies, etc.
#5. Analyze the code used for the renderTable function.
    # Notice where each bit of code comes from in the UI and server.
```

```
    # Note: renderTable doesn't work well with dates. "sampledate" appears as # of days since 1970.
```